IN THE UNITED STATES DISTRICT COURT
FOR THE DISTRICT OF DELAWARE

SRI INTERNATIONAL, INC.,
a California Corporation,

Plaintiff and
Counterclaim-Defendant,

v.

INTERNET SECURITY SYSTEMS, INC.,
a Delaware corporation,
INTERNET SECURITY SYSTEMS, INC.,
a Georgia Corporation, and
SYMANTEC CORPORATION,
a Delaware corporation,

Defendants and
Counterclaim- Plaintiffs.

Civil Action No. 04-CV-1199 (SLR)

FILED UNDER SEAL

DECLARATION OF ROBERT M. GALVIN IN SUPPORT OF DEFENDANTS' OPPOSITION TO SRI INTERNATIONAL, INC.'S MOTION FOR PARTIAL SUMMARY JUDGMENT OF NO ANTICIPATION BY COMBINATION OF REFERENCES

Richard L. Horwitz (#2246)
David E. Moore (#3983)
POTTER ANDERSON & CORROON LLP
Hercules Plaza, 6th Floor
1313 N. Market Street
Wilmington, DE 19899
Tel.: (302) 984-6000
Fax: (302) 658-1192

OF COUNSEL:
Holmes J. Hawkins III
Natasha H. Moffitt
KING & SPALDING LLP
191 Peachtree St.
Atlanta, GA 30303
Tel: (404) 572-4600
Fax: (404) 572-5100

Theresa A. Moehlman
KING & SPALDING LLP
1185 Avenue of the Americas
New York, New York 10036
Tel.: (212) 556-2100
Fax: (212) 556-2222

Attorneys for Defendant
INTERNET SECURITY SYSTEMS, INC.,
a Delaware Corporation and
INTERNET SECURITY SYSTEMS, INC.,
a Georgia Corporation

Richard K. Herrmann (#405)
Mary B. Matterer (#2696)
MORRIS, JAMES, HITCHENS
& WILLIAMS, LLP
222 Delaware Avenue, 10th Floor
Wilmington, DE 19801
Tel: (302) 888-6800
Fax: (302) 571-1750

OF COUNSEL:
Lloyd R. Day, Jr. (*pro hac vice*)
Robert M. Galvin (*pro hac vice*)
Paul S. Grewal (*pro hac vice*)
DAY CASEBEER MADRID
& BATCHELDER LLP
20300 Stevens Creek Blvd., Suite 400
Cupertino, CA 95014
Tel: (408) 873-0110
Fax: (408) 873-0220

Michael J. Schallop (*pro hac vice*)
Symantec Corporation
20330 Stevens Creek Blvd.
Cupertino, CA 95014
Tel: (408) 517-8000
Fax: (408) 517-8121

Attorneys for Defendant
SYMANTEC CORPORATION

Original Date: June 30, 2006

Redacted Date: July 7, 2006

IN THE UNITED STATES DISTRICT COURT
FOR THE DISTRICT OF DELAWARE

SRI INTERNATIONAL, INC., a California Corporation, Plaintiff and Counterclaim-Defendant, v. INTERNET SECURITY SYSTEMS, INC., a Delaware corporation, INTERNET SECURITY SYSTEMS, INC., a Georgia Corporation, and SYMANTEC CORPORATION, a Delaware corporation, Defendants and Counterclaim- Plaintiffs.	Civil Action No. 04-CV-1199 (SLR) **FILED UNDER SEAL** THIS DOCUMENT CONTAINS MATERIALS WHICH ARE CLAIMED TO BE CONFIDENTIAL AND COVERED BY A PROTECTIVE ORDER. THIS DOCUMENT SHALL NOT BE MADE AVAILABLE TO ANY PERSON OTHER THAN THE COURT AND OUTSIDE COUNSEL OF RECORD FOR THE PARTIES

DECLARATION OF ROBERT M. GALVIN IN SUPPORT OF SYMANTEC CORPORATION'S OPPOSITION TO SRI INTERNATIONAL, INC.' S MOTION FOR PARTIAL SUMMARY JUDGMENT OF NO ANTICIPATION BY COMBINATION OF REFERENCES

I, Robert M. Galvin, declare that:

1. I am a member of the law firm of Day Casebeer Madrid & Batchelder LLP, counsel for Defendant Symantec Corporation. I am admitted to practice law before the courts of the State of California, and I am admitted *pro hac vice* to this Court for this case.

2. I make this declaration of my own personal knowledge. If called to testify as to the truth of the matters stated herein, I could and would do so competently.

3. Attached hereto as Exhibit A is a true and correct copy of a document authored by Craig Zacker and Paul Doyle, et al, entitled, "Upgrading and Repairing Networks" (Que Corporation 1996).

4. Attached hereto as Exhibit B is a true and correct copy of a document authored by Karen Watterson entitled, "Understanding SNMP," WindowsITPro (May 1997).

5. Attached hereto as Exhibit C is a true and correct copy of selected pages from Computer Dictionary (Que Corporation 1998).

6. Attached hereto as Exhibit D is a true and correct copy of RealSecure, "Market Requirements Documents – September 1997 – v0.2" (September 1997).

7. Attached hereto as Exhibit E is a true and correct copy of a document authored by Terrance Lee Goan Jr., entitled "Towards a Dynamic System for Accountability and Intrusion Detection in a Networked Environment", M.S. Thesis, Division of Computer Science, University of California, Davis, 1992 [SYM_P_0598736-0598795].

8. Attached hereto as Exhibit F is a true and correct copy of the Statement of Work (SOW) for Arca Systems Inc. and Associates Support to Trident Data Systems (July 26, 1993) [SRI 027600-601].

9. Attached hereto as Exhibit G is a true and correct copy of a web page entitled "UC Davis" (April 23, 1996) [SRI 094671-72].

10. Attached hereto as Exhibit H is a true and correct copy of selected pages from NetRanger Manual User's Guide, Version 1.3.1 [SYM_P_0075018-22, 0075028].

11. Attached hereto as Exhibit I is a true and correct copy of selected pages from Managing InternetWorks with SNMP, Second Edition, Miller, Mark A., P.E., 1997 [SYM_P_0504027-29].

12. Attached hereto as Exhibit J is a true and correct copy of Todd Heberlein Notes from February 11, 1992 [HEB_0004376, 4446-47].

13. Attached hereto as Exhibit K is a true and correct copy of Todd Heberlein Notes from March 15, 1994 [HEB_0004252].

14. Attached hereto as Exhibit L is a true and correct copy of selected pages

from the Deposition Transcript of Todd Heberlein.

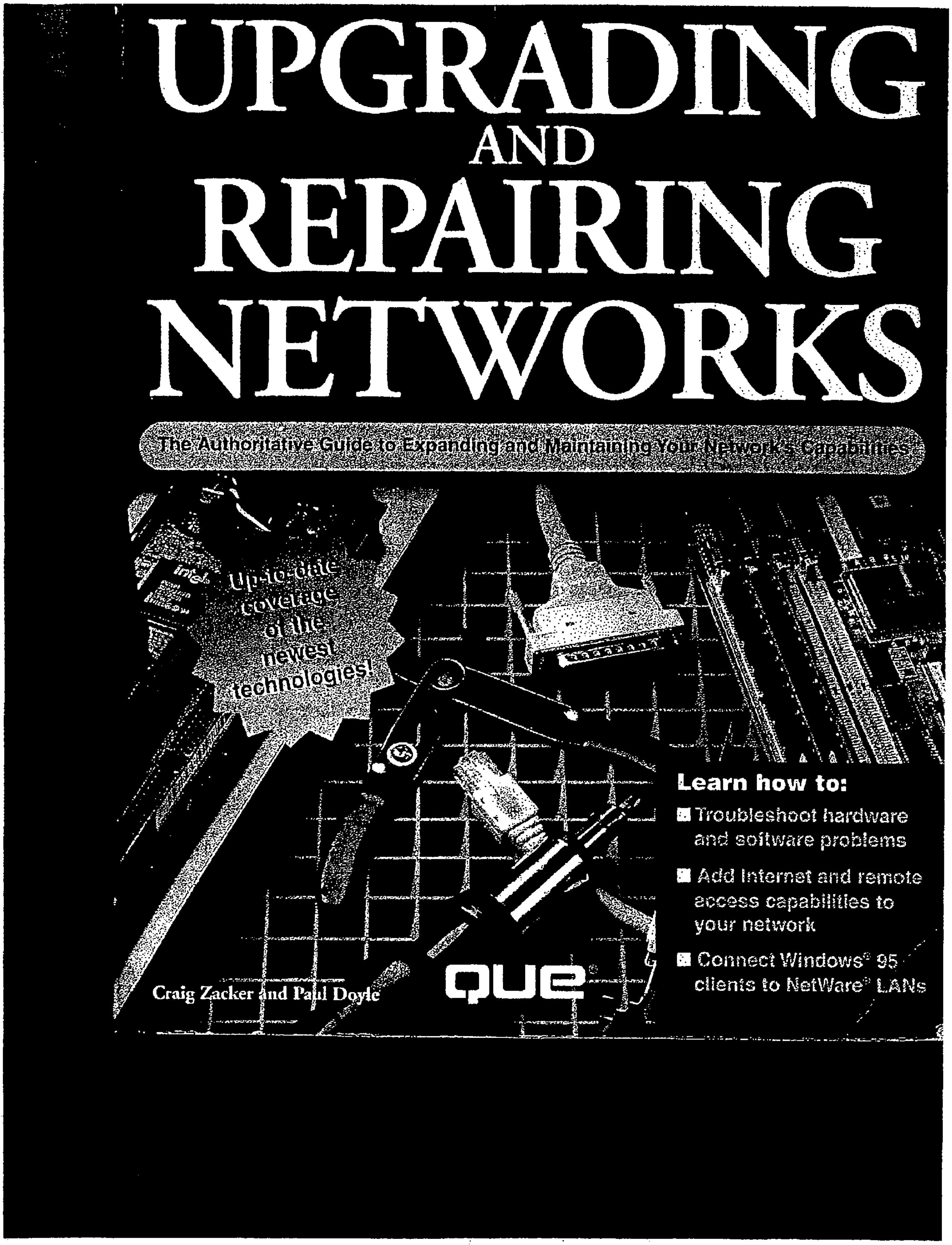
I declare under penalty of perjury under the laws of the United States that the foregoing is true and correct to the best of my knowledge.

Signed on June 30, 2006.

[illegible]

Robert M. Galvin

EXHIBIT A



UPGRADING AND REPAIRING NETWORKS

The Authoritative Guide to Expanding and Maintaining Your Network's Capabilities

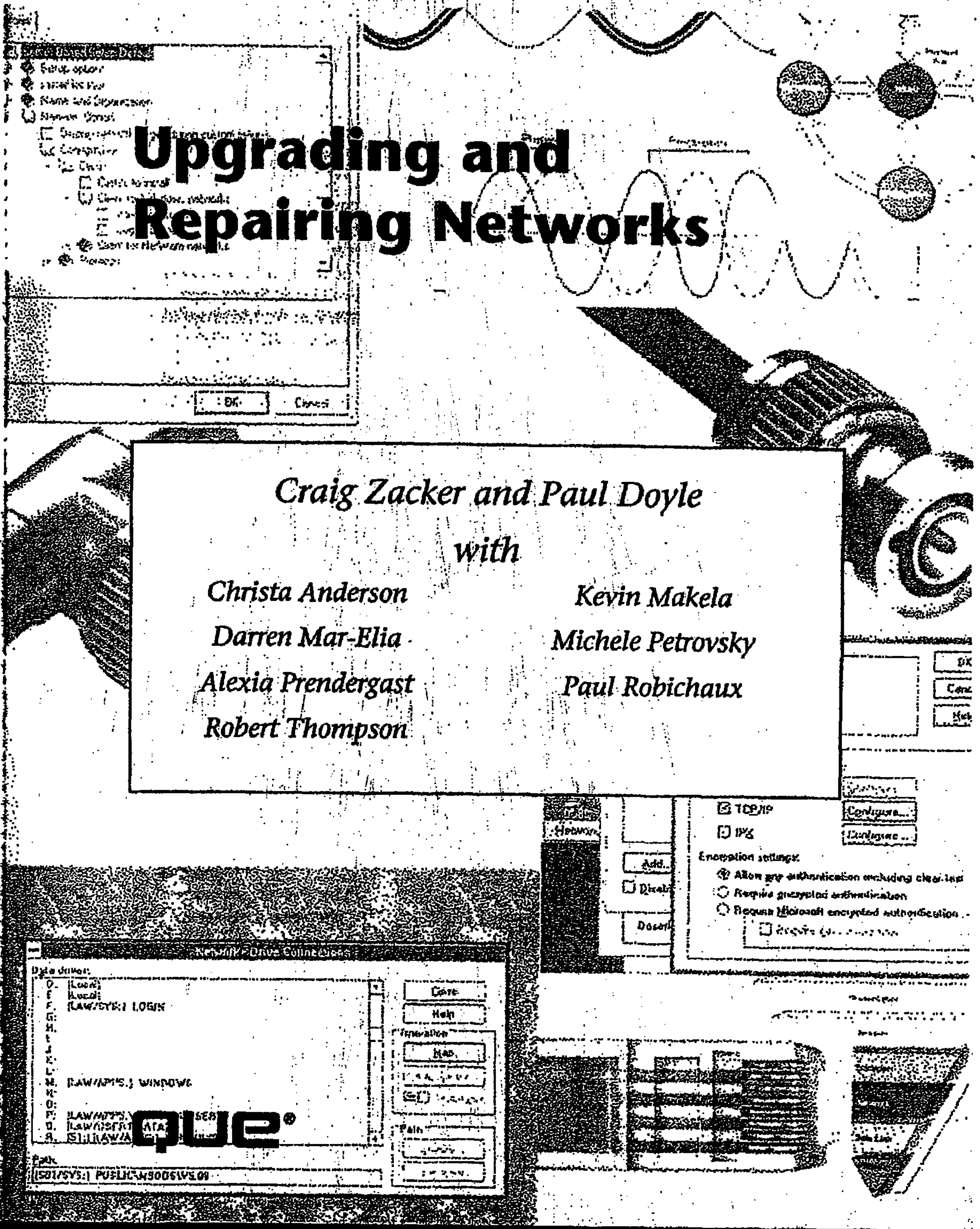
Up-to-date coverage of the newest technologies!

Learn how to:

- Troubleshoot hardware and software problems
- Add Internet and remote access capabilities to your network
- Connect Windows® 95 clients to NetWare® LANs

Craig Zacker and Paul Doyle

QUE®



Upgrading and Repairing Networks

Craig Zacker and Paul Doyle

with

Christa Anderson
Darren Mar-Elia
Alexia Prendergast
Robert Thompson
Kevin Makela
Michele Petrovsky
Paul Robichaux

que®

[redacted]

Upgrading and Repairing Networks

Library of Congress Catalog No.: 95-71740

ISBN: 0-7897-0181-2

98 6 5 4

Interpretation of the printing code: the rightmost double-digit number is the year of the book's printing; the rightmost single-digit number, the number of the book's printing. For example, a printing code of 96-1 shows that the first printing of the book occurred in 1996.

Screen reproductions in this book were created using Collage Plus from Inner Media, Inc., Hollis, NH.

with your NIC vendor before deciding to use the software agent approach. It's also important to realize that the software agent will impose some resource requirements on the server and NIC that may not be acceptable on a very busy system. Finally, any solution, agent or dedicated hardware, will quickly become costly if you try and monitor every segment in your network. Pick the most heavily utilized segments first and then add as needed.

Network-Management Protocols

Depending on your platform, you may be able to implement any number of network-management protocols to support your network. The *network-management protocol* is the method by which your agents and network managers exchange information. The protocol will define what transport mechanisms can be used, what information exists on an agent, and in what format that information is arranged. By far, the most popular protocol for managing networks is the *Simple Network-Management Protocol (SNMP)*. Originally designed for the TCP/IP-based Internet, it has been implemented on other protocols, including Novell's IPX/SPX, Digital's DECNet, and AppleTalk. Since it is based on widely accepted Internet standards, it has a great many advantages and is supported by most, if not all, serious network-management vendors. Because of this role, it is highly recommended that if you are planning a network-management strategy, you focus on SNMP as your protocol of choice. In the long run, it will give you the most flexibility, extensibility, and vendor and end-user support.

Understanding SNMP History

SNMP was actually not the first management protocol defined for use on the Internet. Back in 1987, the *Simple Gateway Monitoring Protocol (SGMP)* was defined by a Request for Comment (RFC) to manage the ever-expanding router network on the Internet.

How the RFC Process Works

[illegible]

In 1989, the first RFC on SNMP received Recommended status, having been built on the experience learned from the short life of SGMP, and the need to manage devices other than just routers. Currently, RFC 1157, written in 1990, is the most recent update to SNMP version 1. Related to the SNMP standards are the associated *Management Information Base-I* and *-II* (MIB-I and MIB-II, respectively) standards, which define the contents of the agent software. If we refer back to our discussions of agents, then the MIB is a specification for what items are managed by the agent software. The MIB attempts to provide a generic list of functions that many network devices have in common. For vendor-specific features, the MIB provides a way for vendors to add on their own.

EXHIBIT B

WindowsITPro
Connecting The IT Community

May 1, 1997

Understanding SNMP

By Karen Watterson, Karen Watterson

Simple Network Management Protocol (SNMP) is a vendor-independent protocol for transporting management data between networked devices and applications and the systems that control and monitor those devices and applications. The Internet community first developed SNMP in 1987 to augment existing TCP/IP network management tools (e.g., ping). SNMP is based on Internet standards. Several Internet Engineering Task Force (IETF) Request for Comments (RFCs) define SNMP (notably RFC 1157). Because of SNMP's origins, many users implement it over IP. In fact, User Datagram Protocol (UDP) ports 161 and 162 identify SNMP agents and managers, respectively. However, you can deliver SNMP messages using other protocols such as SNA.

MIBs, Agents, and Traps

SNMP-compliant hardware and software ship with agent software to help track relevant information about network traffic and device or application statistics. These hardware and software resources store up-to-date information in management information bases (MIBs--see IETF RFCs 1213 and 1573). MIBs are hierarchical name spaces that contain relevant information about the device or application that you are monitoring with SNMP.

Agent software generally waits to be polled by the SNMP-monitoring software and then returns the current values for the requested MIB objects. In addition to receiving information from devices and applications with built-in MIBs, several agent-building toolkits let you define custom MIBs and agents.

SNMP primarily uses two types of commands: *get* (to retrieve information) and *set* (to change it). Another command, *trap*, sends an alarm to a management station under predefined conditions.

SNMP's primary goal is simplicity, so it contains only three *get* commands: GetRequest, GetResponse, and GetNextRequest. A new *get* command, GetBulk (which will let you retrieve information from more than one MIB at a time), is part of the proposed SNMPv2* (V2 star) specification in IETF RFCs 1441 through 1452.

SNMPv2* is emerging to address problems of scalability--because of the nature of the simple manager and agent model, SNMP doesn't scale well--manager-to-manager communications, and security. Be aware, however, that many individuals in the computer industry do not support the adoption of SNMPv2* because it isn't a simple superset of SNMP and it doesn't support backward compatibility. For more information about SNMP, check the following Web sites and USENET newsgroup:

- University of Twente's SNMP site: http://wwwsnmp.cs.utwente.nl
- Douglas Stevenson's overview of network management: http://netman.cit.buffalo.edu/Doc/DStevenson
- LIDO Telecommunications Learning Center: http://www.lidoorg.com/networkmgt.htm
- USENET newsgroup: comp.protocols.snmp

Penton
Where media is going™

Reprinted from Windows IT Pro.

EXHIBIT C

COMPUTER *Dictionary*

Data Communications,
PC Hardware, and
Internet Terminology

Mitchell Shnier

QUE®

Scott Mueller Library
SERIES

Computer Dictionary

Copyright © 1998 by Que® Corporation.

Library of Congress Catalog No.: 98-84693

ISBN: 0-7897-1670-4

00 99 98 6 5 4 3 2 1

Interpretation of the printing code: the rightmost double-digit number is the year of the book's printing; the rightmost single-digit number, the number of the book's printing. For example, a printing code of 98-1 shows that the first printing of the book occurred in 1998.

Screen reproductions in this book were created using Collage Plus from Inner Media, Inc., Hollis, NH.

Executive Editor
Angie Wethington

Development Editor
Jim Minatel

Acquisitions Editor
Stephanie McComb

Managing Editor
Thomas F. Hayes

Cover Designer
Dan Armstrong

Book Designer
Argosy

- *Data Link Switching* (DLSw), which encapsulates an SNA frame into an TCP/IP packet.
- *Dependent LU* Requester/Server DLUR/S, which uses APPN at the transport protocol.
- *Frame Relay*, using encapsulation as defined in RFC 1490.

See S3270, APPC, APPN, DLC, FEP, FRAME RELAY, LU-6.2, MAINFRAME, SDLC, SAA, SNI, TOKEN RING, and VTAM.

SNAFU
Situation Normal: All F_____ Up

Another one of those cynical and colorful terms that was originated by the military during the second world war. It means that things are running as well as they usually do. Unfortunately, many now use the term to indicate a problem.

See FOO.BAR.

SNAP
Subnetwork Access Protocol

A frame format often used for TCP/IP and Apple's EtherTalk on 802.3 ("Ethernet") LANs.

See S802.3.

SNI
SNA Network Interface

The old way of connecting different company's IBM mainframe computers (TCP/IP might be a better way today).

See MAINFRAME, and SNA.

Sniffer
Network General Sniffer

"Sniffer" is the trademarked name for the LAN protocol analyser made by Network General Corp.

Before RMON and powerful PCs which are now able to do a good job of packet capture and analysis, the Sniffer was *the* high-end LAN analysis tool to have. It consists of a powerful portable PC with a custom-designed LAN adapter. The adapter gave the Sniffer the capability of capturing illegally short or long frames (which are typically discarded by standard LAN adapters).

Network General is at *http://www.networkgeneral.com*.

See RMON.

SNMP
Simple Network Management Protocol

A query/command/response protocol to examine and change configuration parameters and counters of LAN- and WAN-connected repeaters, bridges, routers, and other devices.

Agents (software running in the monitored/controlled equipment) communicate with *management stations* (usually using TCP/IP over an Ethernet LAN). Agents store and update their information and parameters as counters, arrays or tables of values. Devices that have such SNMP agents are often called *managed*—such as a *managed bridge* or a *managed repeater*.

Proxy agents are used to manage devices that have a non-SNMP management protocol. For example, a CSU/DSU or modem that has an EIA-232 interface cannot talk TCP/IP over Ethernet (which is the usual way to carry SNMP information). Therefore a PC or stand-alone box (the proxy agent) is used. It talks the proprietary protocol over EIA-232 to the device, and converts this to SNMP and TCP/IP over Ethernet (for example).

The SNMP protocol supports only the following functions (which is why it's called a *simple* network management protocol!):

▲ `Get` (a specified variable's current value from an agent)

▲ `GetNext` (get the next variable's value)

▲ `Set` (a variable)

▲ `Trap` (the agent sends a message when a threshold is exceeded)

All variables are defined by MIBs (*Management Information Bases*), though most equipment has manufacturer-specific extensions. These extensions are described by using a subset of a language called *Abstract Syntax Notation One* (ASN.1), which is described in the *Structure of Management Information* (SMI—see RFCs 1155 and 1212). A *MIB compiler* is then used to integrate the extensions into the management station's SNMP software.

Uses TCP/IP's UDP connectionless transport. (IPX, AppleTalk, and OSI transports have been defined but are seldom used or implemented.) IBM is working on an SNA implementation.

Very little security is defined. All communicating agents are assigned a *community string* (sort of a weak password), which is not encrypted when it is sent over the network. By default, the read-access community string is `public`, and the write-access community string is `private`. These default passwords are a security problem because many installations do not change them. A common work-around that provides a slightly better level of security is that devices are configured (initially, usually from a directly-attached console) so that further configuration changes are only accepted from SNMP management stations with certain IP addresses. The problem with this is that if someone learns the IP address of the management station (not a difficult thing to do), they could program their PC to use that IP address.

A recent extension S-SNMP (*Secure SNMP*) adds security features such as user authentication and data encryption (DES) but adds significant network overhead. This extension has been bogged-down for years in the standards-setting process.

The major UNIX-based management stations (in decreasing order of number of installations and third-party applications available) are:

▲ Hewlett-Packard Co.'s *OpenView Network Node Manager*, running on an HP Apollo (running HP-UX), Sun (running SunOS), or Microsoft Windows platform

▲ Sunsoft Inc.'s *Sunnet Manager*, running on a Sun SPARCstation (used to be the most popular)

▲ IBM's *Systemview* or *Netview/6000* (which is based on HP's Openview), running on an IBM RS/6000 workstation (running AIX), Sun, or Microsoft Windows platform

A problem with SNMP is that it requires continuous polling (which loads the network—especially for larger networks), and most MIBs provide only current information. To get historical information, the management station must continuously poll and archive the current information. RMON is a widely implemented and accepted improvement to this.

SNMP was developed by Jeff Case (along with others), who was a network manager and computer science professor at the University of Tennessee in Knoxville. They developed the management station, and Proteon (a popular Router vendor at the time) developed agent software for their equipment. The first implementation of the protocol was called simple gateway monitoring protocol (SGMP—routers were often called gateways then), and was completed in the summer of 1987. SNMP was accepted as an interim standard for TCP/IP networks in February 1988 (it was expected that CMIP would replace it). SNMP is standardized in RFCs 1098 and 1157.

Information and further links to network management can be found at *http://smurfland.cit.buffalo.edu/netman/index.html*.

See **CMIP, DMI, DMTF, MIB, RMON, SNMP2,** and **UDP.**

SNMP V2
Simple Network Management Protocol version 2

A proposed enhancement to SNMP and Secure SNMP to support larger and faster networks (for example, 64-bit counters are supported) with more complex reporting while causing less network traffic (more than one variable can be retrieved per query, for example).

SNMP v2 can run over AppleTalk, IPX, and OSI transport layer software, in addition to TCP/IP.

Other enhancements include:

▲ A `get_bulk` command is supported, to enable more parameters to be retrieved in a single packet.

▲ Agents can be *locked*, so only one management station can configure it at a time

▲ Better error reporting is supported

▲ A single station can be both a manager and an agent, allowing for hierarchical management

▲ Proxy agents are supported, so one TCP/IP management station can report on many non-TCP/IP agents

▲ DES or public key encryption is used for passwords passed over the network, there are no default passwords, and authentication is supported

Sometimes called *Simple Management Protocol* (SMP).

Originally specified in RFCs 1441 through 1444, which were replaced by RFCs 1902 through 1908.

Major disagreements over the handling of security in the standards development process have delayed the standard, and the availability of products. One group proposed something called *User Based Security* (USEC), and called the resulting standard SNMP v2u. Another group called their version SNMP v2*.

As a result, a less ambitious effort called SNMP v2c was developed that continues to use the basically useless security of SNMP v1, while offering most of the new functionality of SNMP v2. SNMP v2c is specified in RFC 1901 through 1904.

These should all be superceded by SNMP v3, which provides for user athentication and data privacy, as well as proxies to perform polling of data.

A draft of SNMP v3 is at *http://www.ietf.org/html.charters/snmpv3-charter.html.*

See DES, SNMP, and TCP/IP.

SNPP
Simple Network Paging Protocol

A TCP/IP-based protocol to send alphanumeric messages to a *gateway computer* connected to a paging system.

Messages can be up to 900 characters in length.

Support is included for transmitting pagers so that confirmation that the recipient received the page can be sent to the original sender.

See TCP/IP.

SNTP
Simple Network Time Protocol

See NTP.

Sockets

A nonstandardized software interface (API) between a user application program and a TCP/IP protocol stack.

Initially developed (in 1984) for the 4.2BSD version of UNIX. The programming interface is somewhat like that of a file and has the following functions:

- `accept` (an incoming connection)
- `bind` (an address to an outbound call)
- `initiate` (a connection)
- `listen` (for an incoming connection)
- `receive` (a message from a socket)
- `send` (a message to a socket)
- `socket` (create one)
- `close`, `read`, and `write`, too

See API, BSD UNIX, TCP, UNIX, and WINSOCK.

EXHIBIT D

EXHIBIT REDACTED IN ITS ENTIRETY

EXHIBIT E

Towards a Dynamic System for Accountability and Intrusion Detection in a Network Environment

By

Terrance Lee Goan Jr.

Thesis

Submitted in partial satisfaction of the requirements for the degree of

MASTER OF SCIENCE

in

Computer Science

in the

GRADUATE DIVISION

of the

UNIVERSITY OF CALIFORNIA
DAVIS

Approved:

Armand Prieditis.............

Karl N. Levitt.............

[illegible].............

Committee in Charge

1992

DIVISION OF COMPUTER SCIENCE
UNIVERSITY OF CALIFORNIA
DAVIS, CA 95616

i

Acknowledgments

Many thanks to my thesis advisor Armand Prieditis for his advice, comments, and most of all, patience. I would also like to thank Karl Levitt and Biswanath Mukherjee for their suggestions, and for the support and they have given me over the past two years. My thanks also go out to Becky Base, for many thoughtful discussions. I would also like to acknowledge the support of NSA Contract Number MDA904-91-C-7044. Finally, I am extremely grateful for the advice and assistance of all my fellow Netsec-ers.

Abstract

Several well publicized attacks on computer systems (e.g., the Internet worm) have brought increased attention to computer security. With the continual discovery of new system flaws and the threat of abuse by privileged users, there is a need for IDS's (intrusion detection systems). Most of the existing IDS's suffer from two substantial limitations: they do not utilize information about network traffic, and they do not provide facilities for adapting to new attack of computer facilities. In this thesis I present both a method for tracking users across a network, thereby making them accountable for all their actions, as well as machine learning techniques for the on-site modification of intrusion detection expert systems.

Chapter 1 Introduction

Existing IDS's are generally designed for single machines, and utilize little of the information that can be gained from LAN (local area network) activity [12, 18, 20, 24]. This limited view of the computer network environment means that these systems can only be used effectively against a limited range of attacks, and are very susceptible to attackers who use the network to *hide* their activity. The objective of this thesis is to build a system which tracks intruders in a network environment, and improves itself by learning from experience. This work is broken into two pieces. The first is an algorithm which guarantees that all computer system activity will be associated with the appropriate user. And the second is a pair of machine learning techniques which although not yet tested in the IDS domain, show promise in adding flexibility to IDS's.

1.1 Organization

Chapter 2 will provide introductory information on the topics of IDS's, and machine learning, including a review of related research. In chapter 3 DIDS is outlined as a preface to the research done for this thesis, namely the DIDS Expert System, which appears starting in chapter 4 and comprises the rest of the thesis. This work demonstrates how DRA (distributed recognition and accountability) and machine learning techniques can be used to improve the performance of IDS's.

Chapter 2

Previous Work

2.1 Intrusion Detection Systems

Prevention alone is clearly not sufficient to protect computer facilities, since new system vulnerabilities are being continually discovered, and known flaws often go uncorrected due to lack of information or simply the inconvenience that may be caused by such a fix. And even if computer systems were devoid of such security flaws, they would still be susceptible to abuse by privileged users and "masqueraders" who have stolen or broken the passwords of legitimate users. Therefore, there is a dire need for IDS's (intrusion detection systems) which can be used in concert with preventive measures.

The problem of intrusion detection is to detect and explain the unauthorized use, misuse, and abuse of computer resources. These "intrusions" can be originated by a computer "cracker," or by persons who have legitimate access to the computer system (i.e. the insider threat), and can either be a violation of an explicit site policy or general standards of use. To solve this problem, IDS's use *statistical anomaly detection* and/or *rule based attack detection*. While the rule based approach attempts to describe known forms of attack explicitly, anomaly detection systems work under the assumption that "intrusions" can be detected by looking for statistically significant deviation from a user's previous behavior [3, 12]

The anomaly detection systems either generate profiles of a user or groups of users based on several different statistics, or they use inductively generated sequential patterns to characterize normal behavior. Both of these methods make two assumptions. First, that during the training periods, (which are frequent or continuous) "intrusive behavior" is not occurring. Clearly any such activity would corrupt the resulting profiles. Second, these systems assume that "intrusive behavior" is anomalous. That is to say, "intrusive behaviors" differ more significantly form the "norm" than do the day to day variances of use.

Rule based attack detection systems are built by encoding expert knowledge about "intrusions" into an expert system. As with other expert systems, these may suffer from rigidity. That is, these systems will detect *only* the attacks that have corresponding rules in the expert system.

The following is a sampling of existing intrusion detection systems, some using anomaly detection, other using expert systems, and yet some others that use a combination of the two.

IDES

IDES (Intrusion Detection Expert System) is an intrusion detection system developed by SRI International. It utilizes sophisticated statistical algorithms to transform user activity into a "profile" [12].This profile consists of the frequencies, means and covariances of a large group (40) of measurements some of which are:

CPU Usage (sec): The delta of CPU usage between audit records.

I/O Usage (sec): The amount of I/O activity while user is logged in.

Physical Location of User: Where the user logs in from.

Day of Use: Whether the user has generated audit records on a particular day of the week.

An alert is raised if the point in N-space described by the profile vector is sufficiently far from the user's perceived normal behavior. IDES allows any of the profile measures to be turned off if they don't prove useful at a particular site, but it does not have facilities for easily adding new attributes which may be necessary.

The other component of IDES is the expert system (albeit a subordinate component). IDES references (e.g., [12]) do not provide many specifics on their expert system...only that it includes knowledge from past intrusions, known vulnerabilities and site specific security policies.

IDES also attempts to track users across the network. However, their tracking system only works when users do not change user account names, which limits significantly its utility.

Wisdom and Sense

Wisdom and Sense (W&S) uses statistical methods which are used to build a profile of usage patterns on a single host [23]. By using past audit logs to form an instantiated rule forest, W&S generates a rule base of some 10^4 to 10^6 rules. Grades are assigned to these rules according to their rate of occurrence or absence of contrary evidence (noise). The higher the grade, the higher the perceived quality of the rule. Many of the rules that are generated seem to be irrelevant, but some seem to be quite good. Some examples are: highly privileged users don't stay logged in for more than 12 hours, and dial-up users don't stay on for more than 8 hours.

Since W&S is designed for just a single host, it cannot correlate data from attacks which originate from remote hosts. In addition, like IDES, this system does *not* provide facilities for adding new system or user attributes.

Midas

Midas (Multics Intrusion Detection and Alerting System) is designed to work on a single mainframe machine and focuses in on known attack detection [18]. Its rules have been developed by computer security experts and is broken up into 5 areas: break-in, masquerader, penetration, misuse, and Trojan horse/virus. The output of these sets of rules is fed into a second level of rules which determines what action to take. The rules contain information about both user and system anomalies, which help in detecting denial of service attacks and masquerades. The user profiles are simple and include information such as normal times and location of login.

HAYSTACK

HAYSTACK is a single host based IDS which uses statistically analysis to form "suspicion quotients" [20]. These quotients measure the degree to which a user's activity matches the profile of one of six intrusions (browsing, leakage, malicious use, mobility, paranoia, and security penetration). These profiles are formed by processing a large volume of audit trail data and forming a set of "acceptable" ranges for a set of about two dozen features. The acceptable range is computed so that no more than 90% of observed sessions fall within that range. HAYSTACK also concerns itself with changes to access controls and user ids, as well as accesses to objects (e.g., files) that it has "tagged" as requiring special monitoring.

HAYSTACK also includes an attack detection rule base which they call a "signature analyzer" [21]. This component uses fsa's (finite state automata) to represent several known attack forms (e.g., the installation of a Trojan horse program). The states within these fsa's represent various "contexts," and the transitions between states represent particular events.

NSM

The Network Security Monitor (NSM) designed at UC Davis is the only system discussed so far that monitors network traffic over a LAN [8]. It utilizes a simple rulebase which does anomaly and attack detection. The NSM is able to look for events such as connections between hosts that have never communicated before, and multiple failed logins, as well as sensitive material being transferred over the network. By looking at the data being transmitted (assuming no encryption) the NSM can check for strings like "passwd" and "login incorrect" which are suspicious.

Since the NSM relies solely on network activity, it has a few limitations. First, if network traffic is high the NSM may not be able to view every packet, therefore some suspicious strings may go undetected. And second, attackers can hide the use of commands that may appear suspicious by aliasing those commands.

2.2 Machine Learning & Expert Systems

Machine based inductive learning systems learn general rules for classification from a set of specific training examples. Many inductive learning techniques have been developed (e.g., Bayesian classifiers, linear discriminants, neural networks etc.), and used in a wide variety of applications, from pattern recognition to medical diagnosis [23]. The focus of our research has been on decision trees, because they have been shown to perform very well relative to other learning techniques, with the added benefit of representing the product of their learning as a set of easily understood rules which can be used in an expert system [7, 13, 15].

2.2.1 Decision Trees

Decision trees consist of internal nodes (attributes), branches (attribute values), and leaves (classes of objects). Each leaf represents a mutually exclusive subset of the training examples. Figure 4 shows the decision tree generated from the training examples in Table 1. In this hypothetical training set, let "-" equal "bank robber" and "+" equal "responsible citizen."

Training Instances			
Hair Color	**Eye Color**	**Height**	**Class**
Blond	Blue	Short	"+"
Blond	Brown	Tall	"-"
Red	Blue	Tall	"+"
Dark	Blue	Short	"-"
Dark	Blue	Tall	"-"
Blond	Blue	Tall	"+"
Dark	Brown	Tall	"-"
Blond	Brown	Short	"-"

Table 1

Training examples for an unspecified classification task

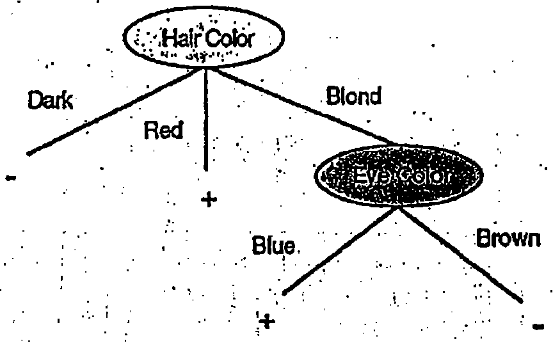


Figure 4. Decision Tree

The tree in Figure 4 is essentially a set of rules which can be applied to classifying new examples. The rules from figure 1 are:

Rule 1: Any new instance with "Dark" hair is classified as a "-"

Rule 2: Any new instance with "Red" hair is classified as a "+"

Rule 3: Any new instance with "Blond" hair and "Blue" eyes is classified as a "+"

Rule 4: Any new instance with "Blond" hair and "Brown" eyes is classified as a "-"

Notice that height is not needed in the classifying task. That is to say, hair and eye color provide enough information to distinguish between bank robbers and responsible citizens.

Decision trees are built from a set S of training examples each assigned to a particular class (e.g., bank robber, responsible citizen) by the following algorithm:

1. If all the examples share the same class name, the decision tree has the value of that class (this is a leaf).

2. Otherwise, choose an attribute A_1 (e.g., hair color) and make that the root of the tree. Partition the examples depending on their value of attribute A_1. These subsets of training examples are then used to form the subtree below the corresponding branch, by repeating step 1.

The method by which the attribute is selected to form internal nodes will have a significant impact of the size of the tree. That is, if attributes are chosen incorrectly then the tree may be quite large (see Figure 5). On the other hand, if attributes are chosen based on their ability to distinguish between classes, then the tree will be smaller and will most likely have a lower error rate when classifying new instances [17].

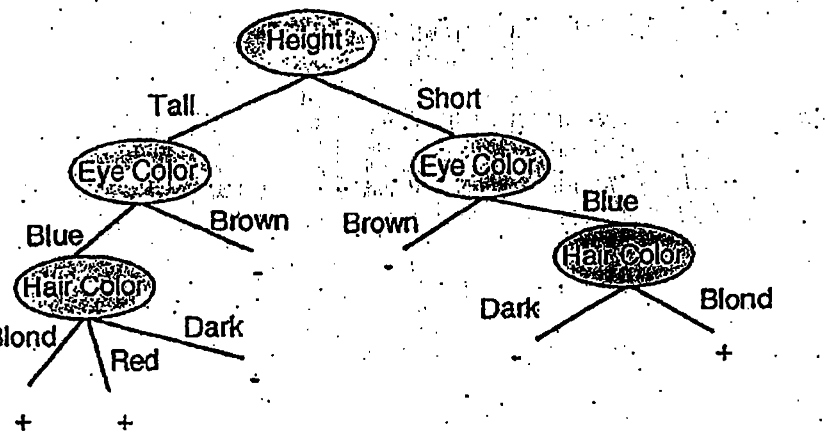


Figure 5. Decision Tree with Poorly Ordered Attributes

Many different heuristics for selecting the attribute on which to partition have been devised. Some are based on entropy or information content, while others are based on error reduction [17]. The most commonly used method is the information gain heuristic which is given below. This method chooses attributes based on the information they provide regarding class membership of the training examples. For instance, the information content of the attribute "height" from above is 0.0016, while the information content of "hair color" is 0.1368 (as calculated below). Clearly "hair color" is the better choice for the root of the tree as can be inferred from the above figures.

p = number of positive instances
n = number of negative instances
p_{ij} = number of positive instances with value v_{ij} of attribute a_i
n_{ij} = number of negative instances with value v_{ij} of attribute a_i

$I(x,y) = - x/x+y \log x/x+y - y/x+y \log y/x+y$

$E(a_i) = \Sigma (p_{ij} + n_{ij})/(p+n) \, I(p_{ij}, n_{ij})$ over all j

Information content $= -(p/p+n \log p/p+n) - (n/p+n \log n/p+n) - E(a_i)$

Although ranking by information content does not guarantee the smallest tree (i.e., shortest mean search path or smallest number of nodes), it has been shown to work well in practice. Actually, the problem of finding minimal decision trees has been shown to be NP-complete [10].

The ID3 algorithm is considered the standard method for building decision trees [14]. ID3 is essentially the algorithm which is described above. ID5R, a modified version of ID3, allows the training instances to be utilized in an incremental fashion [23]. But as with ID3, the vectors used by ID5R are expected to have the same number of attributes.

Requiring that training examples have the same number of attributes amounts to assuming that the initial set of attributes completely describe the items being classified. This is an unrealistic assumption in many applications for which the attributes may not be known (e.g., computer security).

An empirical study of several different methods for handling unknown attribute *values* showed that it is not uncommon in real-world situations for training examples to occasionally be missing attribute *values* (from here on, this will be called the "unknown value" problem) [16]. Quinlan also demonstrated several methods for filling in the missing values by making educated guesses based on the values which were present. The main difference between his study and the problem of adding new attributes is that his missing values appeared more or less randomly throughout the training set, while introducing new attributes will cause a step pattern of missing values (see Figure 6).

"Unknown Value" attribute vectors	Stepwise Introduction of Attributes
[A, 1, True, 23, blue, high]	[A, 1, ?, ?, ?, ?]
[B, 0, True, ?, brown, low]	[B, 0, ?, ?, ?, ?]
[A, 0, ?, 90, ?, low]	[A, 0, False, ?, ?, ?]
*	*
*	*
*	*
[B, 1, ?, 45, blue, high]	[B, 1, True, 45, ?, ?]
[A, ?, False, 23, brown, low]	[A, 0, False, 23, ?, ?]
[A, 0, True, ?, brown, low]	[A, 0, True, 88, blue, High]
*	*
*	*
*	*

? = missing value

Figure 6
Comparison of the "Unknown Value" and "Unknown Attribute problems

Quinlan did not study the effects of such a step pattern nor the impact of introducing attributes in different orders. In addition, Quinlan restricted his study to categorical data. Section 5.1 demonstrates that decision trees which allow for *incremental introduction of attributes* (categorical or <u>continuous</u>) out-perform those that start training from scratch, or those that continue training with a poor set of attributes.

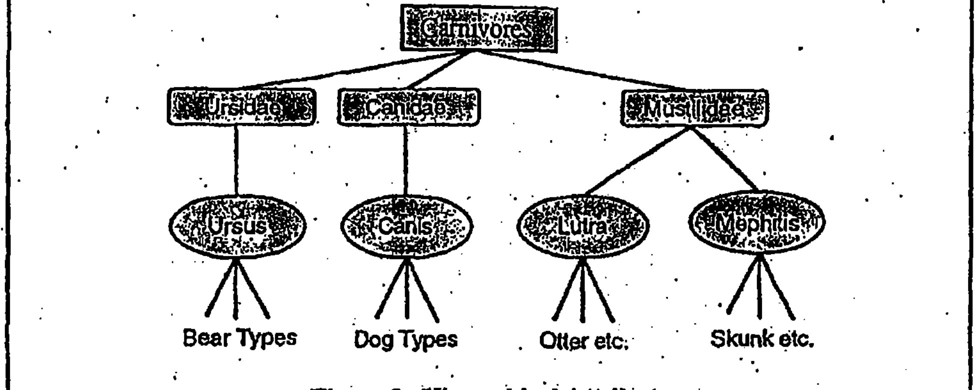


Figure 2. Hierarchical Attribute

Another concept that should prove useful in intrusion detection systems is hierarchical attributes. Figure 2 gives an example of a hierarchical attribute called "Carnivores." All the descendants of this attribute represent the values which it can take on. As you can see, all children are sub-concepts of their parent. So, for example, if all "intruders" are skunks or otters, then you can generalize to "all intruders are mustilidae." It is important to note that traditional decision tree algorithms *do* allow attributes to take on all the values shown in Figure 2, but they are all at the same level (see Figure 3). This is hardly acceptable since attributes with large numbers of values cause decision trees to perform poorly [17]. Section 5.2 will also discuss another useful result of using hierarchical attributes, which is the ability to *control* the generality of a concept which is to be learned.

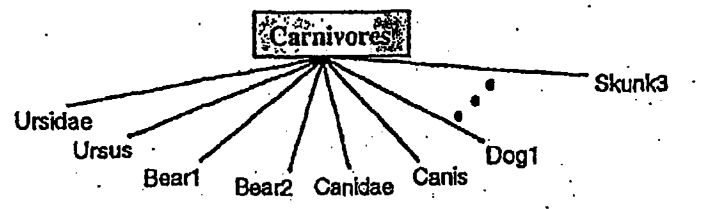


Figure 3. Flat ID3 Attribute

2.2.2 Expert systems

An expert system is a computer program that attempts to approximate the decision making ability of a human expert. Shapiro [19] suggests that the structure of an expert system be split into three distinct modules: the inference engine, knowledge base, and the knowledge acquisition module. The knowledge base contains a set of rules which represent the human expert knowledge. The inference engine determines, given a set of "facts," which rule should be applied next (and then applies that rule). And finally the knowledge acquisition module is used as an aid in extracting the expert knowledge (i.e., the knowledge base) from a human expert.

Knowledge acquisition is generally a very difficult task since most of an expert's decision making process is not conscious [19]. To aid in the task of building the knowledge base Shapiro developed an interactive version of ID3. This systems displays a set of rules which have been derived so far, and the human expert can guide the application of new training examples as he sees necessary. That is, he can choose examples which may clear up rules which he sees as being incorrect. By this method the expert system is tuned until the expert is satisfied with the resulting set of rules.

Section 5.3 discusses an extension of the work done by Shapiro which utilizes *stepwise introduction and deletion of attributes* in order to give the human expert more freedom in tuning expert system rules derived from ID3.

Chapter 3

The Distributed Intrusion Detection System

The U.C. Davis Distributed Intrusion Detection System (DIDS) is designed to automatically monitor multiple hosts connected via a network, as well as the network itself, for misuse or attack [2]. Currently DIDS monitors a collection of Sun workstations interconnected by an Ethernet local area network (LAN). The DIDS components include the DIDS Director, a single Host Monitor per monitored host, and a single LAN Monitor for each LAN segment of the monitored network. The information gathered by these distributed components is transported to, and analyzed at, a central location (viz. an expert system, which is a sub-component of the director), thus providing the capability to aggregate information from different sources. Generalizations to the above platform are planned such as incorporating new host types (e.g., VMS-based workstations) and expanding the monitored domain to wide area networks.

One of the more interesting challenges for intrusion detection in a networked environment is to track users and objects (e.g., files) as they move across the network. For example, an intruder may use several different accounts on different machines during the course of an attack. Correlating data from several independent sources, including the network itself, can aid in recognizing this type of behavior and tracking an intruder to his source. In a networked environment, an intruder may often choose to employ the inter-connectivity of the computers to hide his true identity and location. A single intruder may use multiple accounts to launch an attack, and such behavior can be recognized as suspicious only if one knows that all of the activity emanates from a single source. For example, it is not particularly noteworthy if a user inquires about who is using a particular computer (e.g., using the UNIX who or finger command). However, it may be indicative of an attack if a user inquires about who is using each of the computers on a network and then subsequently logs into one of the hosts. Detecting this type of behavior requires correlating the activities within multiple sessions, perhaps under different account names, to a single user. The problem of Distributed Recognition and Accountability (DRA) is to track such a perpetrator through all intermediate hosts and aliases in the network and credit all distributed activity to that user.

Our solution to the multiple user identity problem is to create a network-user identification (NID) the first time a user enters the monitored environment, and then to apply that NID to any further instances of the user. All evidence about the behavior of any instance of the user is then accountable to the single NID. Next, data must be collected from multiple hosts and LAN Monitors, and correlated such that distributed user activity are accounted to a network-wide user identity.

Inherent in the distributed computing environment are questions of reliability. First, current auditing capabilities are lacking in both detail and scope of the information they provide. Our experience with DIDS has exposed large gaps in the area of system auditing which we discuss in the section on DRA. Second, since DIDS operates within the same unprotected network that it monitors, we have developed several heuristics to cope with realistic network conditions such as packet delays, lost packets and race conditions. Finally, I propose techniques to deal with hosts that provide a restricted amount of information about remote connections (e.g., unmonitored hosts).

The next section presents some scenarios of intrusions and attacks which provide motivation for the network-based approach to intrusion detection.

3.1 Scenarios

The detection of certain attacks against a networked system of computers requires information from multiple sources. A simple example of such an attack is the so-called doorknob attack. In a doorknob attack the intruder's goal is to discover, and gain access to, insufficiently-protected hosts on a system. The intruder generally tries a few common account and password combinations on each of a number of computers (see Figure 7).

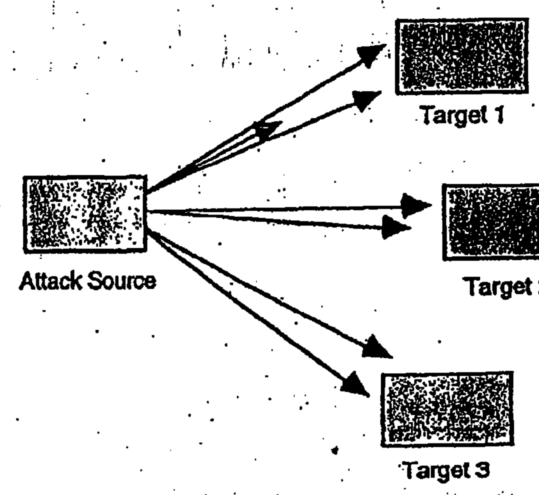


Figure 7. Doorknob Attack

These simple attacks can be remarkably successful [11]. For example, U.C. Davis' NSM recently observed an attacker of this type gaining super-user access to an external computer which did not require a password for the super-user account. In this case, the intruder used telnet to make the connection from a university computer system, and then repeatedly tried to gain access to several different computers at the external site. In cases like these, the intruder tries only a few logins on each machine (usually with different account names), which means that an IDS on each host may not flag the attack. Even if the behavior is recognized as an attack on the individual host, current IDS's are generally unable to correlate reports from multiple hosts; thus they cannot recognize the doorknob attack as such. Because DIDS aggregates and correlates data from multiple hosts and the network, it is in a position to recognize the doorknob attack by detecting the pattern of repeated failed logins even though there may be too few on a single host to alert that host's monitor.

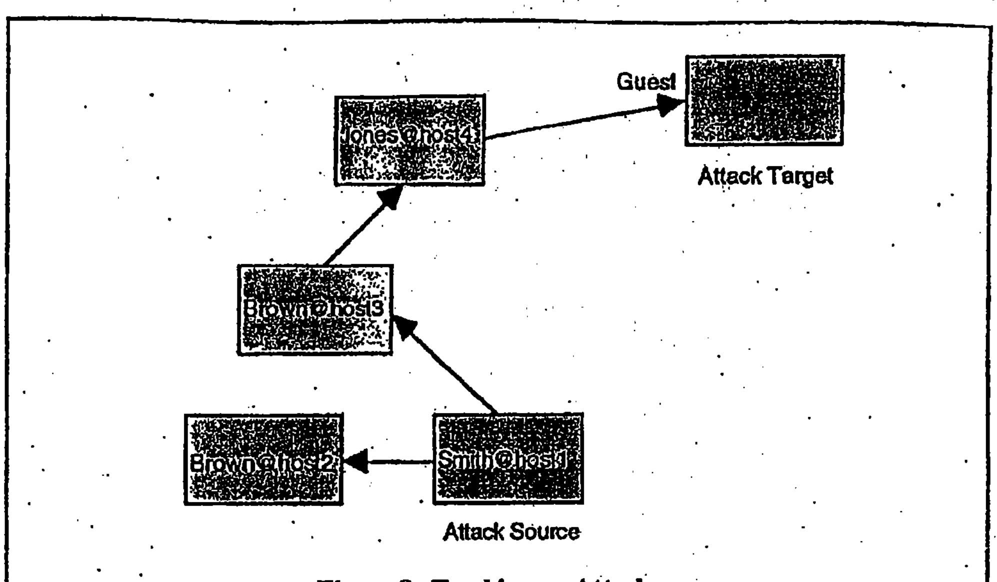


Figure 8. Tracking an Attacker

In another incident, our NSM recently observed an intruder gaining access to a computer using a guest account which did not require a password. Once the attacker had access to the system, he exhibited behavior which would have alerted most existing IDS's (e.g., writing to sensitive files). In an incident such as this, DIDS would not only report the attack, but may also be able to identify the source of the attack (see Figure 8). That is, while most IDS's would report the occurrence of an incident involving user "guest" on the target machine, DIDS would also report that user "guest" was really, for example, user "smith" on the source machine, assuming that the source machine was in the monitored domain. It may also be possible to go even further back and identify all of the different user accounts in the "chain" to find the initial launching point of the attack.

In addition to the specific scenarios outlined above, there are a number of general ways that an intruder can use the connectivity of the network to hide his trail and to enhance his effectiveness. Some of the attack configurations which have been hypothesized include chain and parallel attacks [6]. DIDS combats these inherent vulnerabilities of the network by using the very same connectivity to help track and detect the intruder.

3.2 DIDS Architecture

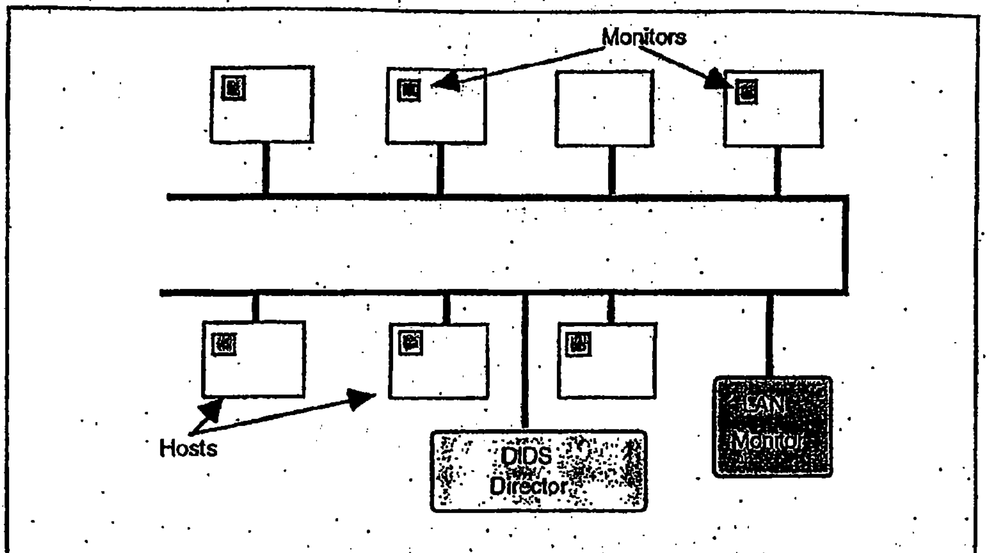


Figure 9. DIDS Architecture

DIDS is designed to operate in a heterogeneous environment composed of C2 [5] or higher rated computers. The current target environment consists of several hosts connected by a broadcast LAN segment (presently an Ethernet, see Fig. 9.). DIDS employs Sun Microsystem's Basic Security Module (BSM) or Digital Equipment Corp.'s VMS auditing depending on the host type. The use of C2 rated systems implies a consistency in the content of the system audit trails. This allows us to develop standard representations into which we can map audit data from UNIX, VMS, or any other system with C2 auditing capabilities. The C2 rating also guarantees, as part of the Trusted Computing Base (TCB), the confidentiality and integrity of the host's audit records.

The DIDS architecture combines distributed monitoring and data reduction with centralized data analysis. This approach is unique among current IDS's. The components of DIDS are the DIDS Director, a single Host Monitor per host and a single LAN Monitor (a subset of the NSM discussed above) for each broadcast LAN segment in the